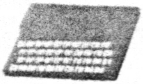
Amateur Publication of—
William Harmer
97 RUSKIN STREET
OTTAWA, ONTARIO
K1Y 4B3



TS Bulletin #2-91

Ottawa TS Support Unit, Since 1987
---This Issue --8 pages--

--Sept. 1991 ---Amateur Publ. © 1991 in the public domain unless otherwise marked-
-- photocopy freely -- for non-commercial use only --

This issue rounds out our fifth year of publication, continuing to "survey... the prospects and potential applications of Timex-Sinclair line of computers...." as said in our FIRST ISSUE in January 1987. If there were an award for longevity perhaps now would be the time to celebrate it. That first issue was four pages. Now we produce 8 page issues although less frequently and put the shorter issues into "News Supplement" newsletters. This issue sees some material surface that was written for a #2-1990 issue that never appeared as the Ottawa-Hull user group was closing down, questioning the reason for our going on. But going on anyway was the eventual conclusion. Two of these items, "What Is A Microprocessor Made of?" (p.6) and the diatomic pack, text data compression algorithm, a continuation of the article in #1-90 TS Bulletin, (p.2) should prove welcome if somewhat overdue in the publication schedule.

Continuing our comitment to hardware hacking is the article on p.7 on Graphics and sophistication that puts some new twists into that topic and some fresh ideas. Also included are some news and address/ supplier updates.

The Sinclair is not dead. I am still programming the ZX-81/ TS1000 in BASIC and incresingly trying out machine language. For that latter the ZX-81/TS1000 is a most excellent practice set-up and a lot cheaper than any other commercial machine language development kit. Indeed, Hacker Electronics selling ZX-81/TS1000 boards for $2 tested ($1 untested) as at Sept., 1991 has made their price ridiculously low. The power supply is not included but plans to make your own out of parts from Radio Shack have been published here before. Moveable-key keyboard adaption instructions are obtainable from any user group still functioning, the surplus keyboards selling for $10 or even less (before adaption).

Well I can't stay at this typewriter much longer with the fall computer weather approaching. I must move over to the desk where the TS1000 awaits. Happy Hobby!

Note: This issue features still more material of use to the classic/ orphan computer user on writing data compression routines, since we are neglected by the big boys who provide ARC & ZIP to the many, and must do our own mostly.

CONTENTS:

-Secrets of Data Compression: A Programmer's Brainstorming Guide . p.4
-Diatomic Pack-Text Data Compression, Part 2 . . . . p.2
-What's A Microprocessor Made Of? (Hardware) p.6
-Cartoons, Computers; Graphics: What They Say And How p.7
-Creative Programming: BASIC, Amateurs And Imagination p.3
- Computer History File. p.5
- Hardware Basics-Disk Drive Unit Electronics p.8

Note: * * * Permission to photocopy/ republish this newsletter in its entirety (only) granted to any existing (now) Timex-Sinclair User Group -Issue #2-1991

Updates, News, Fixes: -Hacker Electronics, 21016 Devonshire, Chatsworth, Ca. USA 91311

- Larken Electronics still has a stock of TS2068 disk interface boards so that orders can still be filled for new LKDOS systems through Larken or RMG Ent.
- Ottawa-Hull TSUG though now defunct has still its individuak TS users in Ottawa continuing to use the TS computers as a hobby. This newsletter has no connection to OHTSUG..

Where This Newsletter Comes From: A labour of love, a sop for idleness, a genuine attempt to HELP other users in their hobby by talking about what I have learned or am thinking about. Editor/ Publisher/ author: B.J.Harmer, 97Ruskin, Ottawa, Can. K1Y 4B3.

Warning: Don't modify or build or pay or make any comitment based on anything in this newsletter without verifying it with another, qualified source.—& Enjoy!

Set up: take one character oot of the text to check, and LET A$= that char.

```
                    POKE 16516, CODE A$
                    LET C=USR 16514
IF C<>8  THEN  LET O$=O$ + CHR$(C .... and a formula, quite complex in the loop
                              to build the double letter codes from octal C for
                              one letter)
IF C=8 THEN LET O$=O$ + A$      --that is not a codable char, so just pass orig.
                                   character through into output unchanged
```

```
                 ---the m/c routine at 16514 is--- (42 bytes long)
                              ;DIATOMIC pack routine, m.c. subroutine part of
hex m/c                       ; assembly language as per Artic Assembler (w. chgs.)
  18 01                         JR START         ; skip over next temp. storage byte
  00              TEMP          0                ; make a temporary storage byte resv.
  11 26 00        START         LD DE,26h        ; load E with char. code 26 hex for"A"
  01 00 00                      LD BC, 0         ; but first octal code, zero in C
  3A 84 40        LOOK          LD A, (TEMP)     ; get 1st char. code from 16516
  BB                            CP E             ; compare to char. code in E
  C8                            RET Z            ; exit m/c routine if same code
  13                            INC DE           ;  move to next target char. code
  13                            INC DE           ;  skipping one letter of alphabet
  03                            INC BC           ; move to next octal code
  79                            LD A, C          ; put octal code into A for compare
  FE 05                         CP 5             ; is octal code 5? (past code of this
                                                ;  part of the routine, A,C,E,G,I
  20 F3                         JR NZ, LOOK      ; if not 5 yet, loop again, looking
                                                 ; for next hit in char sequence
  3A 84 40                      LD A, (TEMP)     ; get tested char. back in A, after
                                                 ; having been over-written by LD A,C
  11 38 00                      LD DE, 38h       ; get char. code for S into E
  BB                            CP E             ; is tested char. S ?
  C8                            RET Z            ; if so exit m/c
  03                            INC BC           ; go to next octal code, 6
  13                            INC DE           ; go to next char. code to test for,
                                                 ; for T
  BB                            CP E             ; is the char. being tested T ?
  C8                            RET Z            ; if so exit m/c
  11 00 00                      LD DE, 0         ; test to see if space, code zero
  03                            INC BC           ; if it is space, octal code will be 7
  BB                            CP E             ;  is it space?
  C8                            RET Z            ; if so exit m/c with 7 in BC reg.,
                                                 ; which will become C from BASIC
                                                 ;  LET C=USR 16514 call
  03                            INC BC           ; if not inc BC up to 8, indicating
                                                 ; no hit
  C9                            RET              ; exit m/c routine, compressed codable
                                                 ; character, one of A,C,E,G,I,S,T,
                                                 ; space, not found
; diatom means two letters coded into one byte=atom, decoding routine not given
; m.c. subrt. takes char.code POKED into 16516, returns 8=no hit," "=7,A=0,C=1,..T=6
```

This of course does not represent a full program either BASIC or m/c code to execute this simplified diatomic pack (algorithm 1A) for text file data compression. The text is separated into single characters, A$ and the m/c runs a test to see if the character code (in this example Sinclair ZX-81, with capitals only beginning at 26 hex, in ASCII codes, 'a' at 97 hex), and it will return octal codes, 0 to 7 for A,C,E,G,I,S,T or space or code 8 if the character is not one of the target characters singled out by this algorithm for compression. This algorithm has succeeded in achieving compression of English (and unaccented French) texts to 75% of original size. The BASIC routine, not shown here, will take the two octal codes for two successive characters from the text and make a one-byte code if both are hits.

Creative Amateur Programming In BASIC

BASIC is certainly not dead yet with new versions for Windows and MS DOS itself new to the market this year. The second-mentioned, QBASIC is a BASIC interpreter version of the compiled Quick BASIC and is meant to replace BASICA/GWBASIC of MS DOS before version 5.0. One thing especially of value to the amateur programmer is that with a BASIC interpreter you can simply start in writing the program and experimenting and trying it. We take that for granted as BASIC programmers but not many other languages offer this. Another is that since so much is already in ROM (at least with the older, classic computers with a built-in BASIC, the programs can be short and still do a lot, where a compiled language even a compiled BASIC would need 6-12K more of RAM-filling overhead (as with MS DOS GWBASIC, Quick BASIC, Turbo Pascal, etc.). It is the ease of experimenting that appeals to so many amateurs and the memory frugality also allows little, old hobby computers to do things in 16K or 32, that users of bigger computers would not think possible. The experimentation that BASIC programmers practice is said to make us unsuitable for the more structured behaviour of professional programming jobs, but is that a short-sighted assessment?

A personnel officer in charge of hiring creative people for the high-tech firms said, on educational television, that to find the really creative person when filling a job, that she would go further than asking what creative solution the applicant was especially proud of in the last job or project. She then would ask after hearing the story, what other ways the person could have tried, and ask for several, and how they compared. She said that a lot of people can brag about the solution they found but fewer can talk of the alternative creative solutions, and why to choose among them the one tried. Of course it is these people who have thought out more than one creative solution before implementing one that are the desirable ones for the job.

This brings us back to BASIC programming. Since it is so easy to experiment as amateur programmers in BASIC and we usually face no deadlines, why don't we program like that more often. After you try one solution to a programming problem and get it working, why not turn off the computer and start scribbling code for two or three alternative and equally creative routines to do that. If it is not obvious which is better, perhaps entering and debugging one or two and testing them for memory consumption and execution time (if that is importaant and it usually is) should be your approach. Amateur programming is supposed to be a hobby and fun and what is more fun than being more creative and imaginative? Anyway, we can safely conclude that perhaps the best professional programmers might be drawn from the ranks of amateur BASIC programmers in the future as in the past when so many got their first experience on computers inexpensive enough to own so they could be picked up and used any time that recreational pauses were possible.

Optimization means often speed but not always. And not always the same amount of speed is necessary. Entering a one-line command or yes or no response hardly needs the speed of input routine best designed for touch-typing of lots of text in a word-procesor. Making a simple INPUT custom routine with INKEY$ could give your computer a new, fancy look even in older, slow classics, by allowing, say Sinclair, computers to type/echo commands from anywhere on the screen and do data entry in boxes, just like a "real database". (Sinclair TS1000/TS2068's have their INPUT use an edit line at the bottom of the screen only, not enywhere on the screen like most computers with scrolling or a screen editor).

You can't add strings? Or can you? It might save memory to be able to increment a byte from 0 to 255 (then back to one) in BASIC to save the 5 bytes minumum to store a floating point number. Also it can simply be POKEd into a machine language routine, into the A register at any time, without further ado, (meaning error trapping/bug catching/preprocessing).

```
INC B   could be implemented as   (given the no, is in B$, by B$=CHR$ B)

      10 INPUT B                  40 LET B$=CHR$((NOT (255=CODE B$))*
      20 LET B$=CHR$(B)                     (1+CODE B$))
      30 PRINT B,B$               50 PRINT CODE B$, B$
```

(Note most BASIC dialects use ASC( ) instead of CODE of Sinclair TS1000 and not all BASIC dialects tolerate all possible CHR$(-) values as well.) This allows no negative numbers as input. DEC, (decrement, or minus one), could also be implemented similarly. Not fast but it will save some RAM especially if you use a lot of bytes, maybe even in an array, in your program. (Times=85/½ms.) ½ compared to FP +1

Secrets of Data Compression: A Programmer's Brainstorming Guide

©1991 by Bill Harmer, Ottawa, All Rights Reserved

Data compression is the software or programmer's answer to the question "How can I get something for nothing?", in this case more data in less memory, RAM or disk. This is the first of a several part series to examine some of the techniques currently used. (Don't assume someone does not have the commercial rights to some of these, like the ARC or PKZIP which are protected.)

Method 1-Null supression or RLE encoding. The principle of this is very simple and it seems to be the first system used in computers. If you have, say, 7 spaces in a row in a text, why use one byte, ASCII code 32, each time to code that. Why not just a special character code reserved for that plus the number 7 to indicate, 7 times this character, in this example a space. RLE, Run Length Encoding says, "Why use this only for runs of spaces? Why not any other ASCII (or numeric, binary etc.) character?" So three bytes, the RLE indicating character, the ASCII character that is repeated and the number of times it is repeated make up one type of RLE code. Nearly all data compression systems use RLE coding or Null suppression (if it is "nothing" or "space" only) in addition to their "main" compression system. Question: Can we only profit by encoding the pattern of a series of characters being the same? Are there other patterns that could also be encoded profitably. What about a fourth byte in the RLE code to give the repetition pattern, bit by bit, so 10101010 would give the pattern for every second byte, so that you could code the "attribute byte" in a CGA video file (or that of many other computers TS2068, etc.) to indicate the same colour combination? What other way could one imagine? (refer to "Data Compression", by Held, Wiley Pub. for routines in BASIC for null supression and RLE encoding.)

Method 2-Diatomic Coding. Say that you need not use the codes 128 to 255 in ASCII on say the IBM PC. You reserve those codes, the ones with the most significant bit set (to one) for codes of two letters, diatoms, so that the bits in the byte are used as follows: msb is one if diatom verus ordinary ASCII, next 4 bits code a hex number, 0-15 for the most common letters used in English (or largely French too), "space",etaoinshrdluyc and the last 3 bits can hold an octal number to code the first 8 of that list, 0-7, "space",etaoins. The extra number in the 4-bit sequence can be your RLE code so that you can use RLE with this too (see above). You can get somewhere close to 70% with such a system, getting rid of the nedessity for 30% more memory. The Held book mentioned in 1 also has BASIC routines for this and a machine code routine for CP/M is given in the book "A Programmer's Notebook", by D.Cortesi, Prentice-Hall Gp. Question: If this allows us to code such unlikely two-letter combinations as "ii" and "aa" that rarely or never occur in English (How many people work aardvark, into their text files?), is there a better way of choosing the letters to be coded into one byte? Why stop at diatoms? Why not triatoms, getting three characters in one byte? (Hint, a space might be the best one to use as the third element and it would only take one bit more.)

Method 3-Bit Scavenging. In the last example you have seen that an extra bit may be found in ASCII code in the most significant bit. Another might be found if you substitue a code like CHR$(62) for an indicator that the next character is a capital letter, the characters for letters otherwise assumed to be lower case (small letters) whether coded as capitals ("A") or lower case("a") in fact. This gives us another bit to play with in ASCII, EBCDIC, and many other codes. In a database, we can use the MSB to indicate a space following the letter for which the most significant bit is set (to one). This should save some space in a name and address or mailing list file. Similarly coding the first few characters of the last name, say the first two, gives us 4 possibilities (00,01,10, 11), so no change might mean no title before the name, 01=MR., 10=MS , 11=DR., (00=no title). Question: Can you go so far in such bit scavenging to get a telephone number out of space normally only good for name and address? (ans.Well maybe most of a telephone number.)

Applying bit scavenging to straight text, with the MSbit set to 1 indicating that the next character will be a space, a shift to capital letters that the next character will be an "e" or if both indicators used, the next two characters will be "space"+"e" (very common emding for a word in English or French) you can get even better results than the diatom pack and use very little source code to get it. If you are interested, the writer has some BASIC routines to do this written.

That is a quick run down of the more interesting routines adaptable for the amateur programmer. Next article will feature those used more by the professionals including where to find the source code programs (C language).

9.2.91

History of The Computer

- ancient calculators, possibly not all discovered, ? B.C.
- 1600's, calculators, including one invented by Blaise Pascal
- Charles Babbage designs mechanical computer 1834-1871
- Boolean Algebra invented by Prof. George Boole, 1854 published
- Herman Hollerith develops the punched card of IBM fame 1880's
- electronics begins in earnest with the electronic vacuum tube diode developed by Fleming, followed in 1906 by the first vacuum tube amplifier, the triode by de Forest who fights many legal battles with big companies, not always successfully, to protect his invention
- the flip/flop attributed to Eccles and Jordan circa 1917
- 1939-1945 first computers used in World War II: U.K.,USA,Germ.
- 1947 invention of the first transistor, solid state electronics has its first amplifier (diodes used very early back to the days of crystal radio sets, 1920's)
- 1953 custom-made commercial computers using vacuum tubes
- 1957 mass produced computers using Germanium transistors
- 1959 invention of the integrated circuit
- 1964 third generation of computers use integrated circuits
- 1971 first microprocessor, 4004, 4-bit, used in a calculator
- 1974 MITS Altair home computer (in kit form) with 8080
- 1975 Apple I computer first launched by two youths
- 1977 Radio Shack home computer (TRS-80, Model I) - not a kit!
- 1980 Sinclair ZX-80 home computer marketed by Sir Clive Sinclair
- 1981 Sinclair ZX-81 home computer, soon marketed by Timex under the TS1000 name, first computer under $200
- 1981 IBM PC home computer comes on market with 8088 Intel microprocessor

Home Front

Milestones With the home computer thousands educate themselves at no expense to the taxpayer in computer science and have fun doing it.

Millstone - To the ranks of the sports widow, alone while the spouse watches football on the t.v., add the computer widow.

Milestone-The first marriage counsellor to recommend computers to save a marriage; "When you start to lose your temper, just walk away and go down the basement to the computer room ...."

....

Computer User's Nightmare ...

Computer **Bug Trashes** Favourite Disk

What's A Microprocessor Made Of?

What's a microprocessor made of? You hear people say that a certain microprocessor (like the 8086) has 10,000 transistors inside it, but how to visualize that? A microprocessor or any other VLSI silicon chip (VLSI= very large scale integration, the high density chip used for modern microprocessors) is basically a tiny printed circuit board (say ¼ inch square to ½ inch sqaure), in which the transistors are connected, usually by a metal connection grid or layer, right on the silicon chip on which they were formed.

How did anyone think of this? Well. transistors were once made one at a time (and were originally, in the 1950's very expensive, so much so that the $200 and $500 transistors were packaged in jewelry boxes like a wedding ring). The process developed so that they could be made on a wafer of silicon and next step was discovering they could by a process of a number of steps be formed on the silicon wafer, not one at a time, but a bunch at a time. The silicon wafer was then cut into little squares, one transistor per square. Doing the transistors, say 200 at a time, saved a lot of money for each time you went through the dozen or so steps of making one wafer of transistors, you could end up with 200 rather than one. A little later it occured to people that you need not cut all the transistors apart on the silicon wafer, but could leave say 4 or 6 per wafer and join them up in a little amplifying or digital circuit by forming little resistors and metallic connections (equivalent to wires) right on the surface of the wafer after the transistors were formed (or in some cases as part of the same process). Then when you cut apart the chips, instead of having 200 single transistors, you would have say 40 silicon chips with 5 transistors each and the circuitry linking them to form a little amplifier or flip-flop.

A VLSI chip is one of the results of developing techniques to fit as many transistors in as small a space as possible on the silicon chip integrated circuit. 10,000 transistors is a typical figure. Microprocessors which are in effect most of the central processing unit (CPU) of a small computer, are very complicated circuits that need therefore a many-transistor circuit, and were one of the most powerful applications of this silicon chip integrated circuit technology. Before that, the CPU of a computer would come on a printed circuit board, that would be one of many to plug into a computer mainframe (mechanical and electrical printed circuit board housing). Before any integrated circuits were used in computers, in the days of single transistors, the CPU would of course be made up of a number of printed circuit boards, plugged in to the computer unit. So a microprocessor is a printed circuit board in which the circuit is that (of most of the components, less memory - RAM and ROM memory) of a computer CPU. The earliest microprocessors were not dense enough to qualify as VLSI density, but with the Z-80 and 8088 and so on, VLSI became the standard for microprocessor complexity. It is amazing to see the outline of this micro 'printed circuit board' when you put a silicon chip under the microscope, as you would a fly's eye or a feather. That makes you realize, how microminiaturized an integrated circuit is compared to the regular printed circuit board.

In the attempt to get as many transistors as possible on a silicon chip, a relatively new type of transistor at the time (1970's), was starting to be used more and more, especially for VLSI chips and their predecessor the LSI (less dense, Large Scale Integration) silicon chip. This was the MOSFET, or metal oxide silicon field effect transistor. This was simple to construct on a silicon wafer and smaller. Taking up less space per transistor on the chip, it allowed much denser packing and was used in the LSI and VLSI chips when they were first made. The MOSFET is a different kind of transistor than the originally invented transistor (of the transistor radios of the 1960's to 1970's). The original transistor was the NPN or PNP type (both were used, the difference is mainly in the electrical polarity of the connections, the PNP taking a negative collector, the NPN a positive collector connection). They are still used today both as separate transistors (discrete transistors in the engineer's jargon) and as the transistors inside some integrated circuits (actually many integrated circuits). The MOSFET transistors, come in two genders also, n-channel and p-channel, similar to the NPN and PNP transistor distinction, and they are used as the transistors inside virtually all microprocessors including CMOS ones (CMOS means Complementary pair of MOSFET transistors).

Cartoons, Computers, Electronics, Apple Mac Graphics: What They Say And How

As computer hobbyists, and electronics dabblers, it is interesting to observe that languages and the conventions (and rules and exceptions) apply not only to the languages we humans use but also to say, MS DOS command language. Take a program name, one ending in .EXE or .COM and 'call' it simply by using its name and it will start running right after being loaded in from disk. Using the name of a .EXE or .COM file means, by this convention that you want to run if (and load it from the current disk, A:, B:, or C:, whatever context the beginning of the line sets with its prompt). It then took me a long time to learn that many programs could also be followed by something as complicated as PROGNAM.EXE A:xxx.XXX B:yy.YYY which tells the program to take as input, on drive A: disk file xxx.XXX and output the result of processing it by the program to drive B: and call the output file, " yy.YYY". There are lots of other conventions, like the one if you leave out one of these command parts, like the B:yy.YYY, say, it will plug in a default part of its own choosing, perhaps in this case perhaps, A:xxx.XXX (only the program knows for sure). When you compile a program in CP/M's CBASIC compiler, the COMMAND$ statement in BASIC allows you to pick up and process within your BASIC program those other things that the user might put in the command line beyond the program name, etc. I wish other languages would make it that simple! (Turbo Pascal has PARAMSTR.)

So with BASIC and operating system command languages like MS DOS there are these rules and conventions, although they are only languages by extension of the meaning of the word language. We don't think of graphics languages as being languages and having such conventions too, but they do. Even cartoons have certain conventions, like the balloons that characters talk into. If you didn't know the convention, and the further, fluffy ballons that the characters use when only thinking, joined to them with bubbles rather than a line.... Now that Graphic User Interfaces and symbols are being used, we have a whole new set of conventions, like clicking the mouse, when the arrow is over a file folder and dragging it over to a garbage pail, to indicate that you are going to throw out its contents. The concept of dragging is of course a convention. Others could have been chosen, like a zap button with games! I think that the mapping and designing of effective new graphic conventions will be increasingly important to computer software invention. Some for example have complained that with the Apple Mac, you can't get a message, "Are you sure?" in graphic form, before you do something that may be drastic and result in accidental or at least unintended erasure of data. That symbol could be used for are you sure? (An interabang: the ? and ! superimposed‽)

When as(used/old computer) hackers we get into hardware, we encounter another system of conventions, in evolution, and still somewhat loose, but ones we probably should know how to deal with if we are to write down the results of our work. Using graph paper and drawing our schematics or mechanical sketches always with the lines on the graph paper lines, (any straight, square edge or wire) says, "I am a pro." When you go to produce a final schematic or mechanical layout for publication, or just to keep yourself, another convention which will make your work look professional, is not only using a ruler to get straight and not wavy lines, but keeping lines at right angles, at exactly right angles. Luckily that is not hard to do with a simple drawing board. Rather than make an purchase of an expensive one, a homemade one using a piece of plywood, and gluing a framing or rafter square to it, so that the edges stick out over the board slightly, may be all that is needed. Add some cardboard to build up the space inside the square, and buy a good T-square for $5-15 and you are in business as your own draftsperson. Then observing a few conventions as the one in mechanical drawings that a dotted line shows an edge you cannot see because something else is in the way, and adding arrows along the edges to show the sizes of things, and voilà, semi-professional work, rather than amateurish!

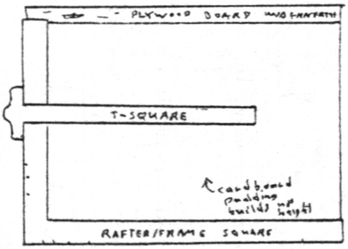
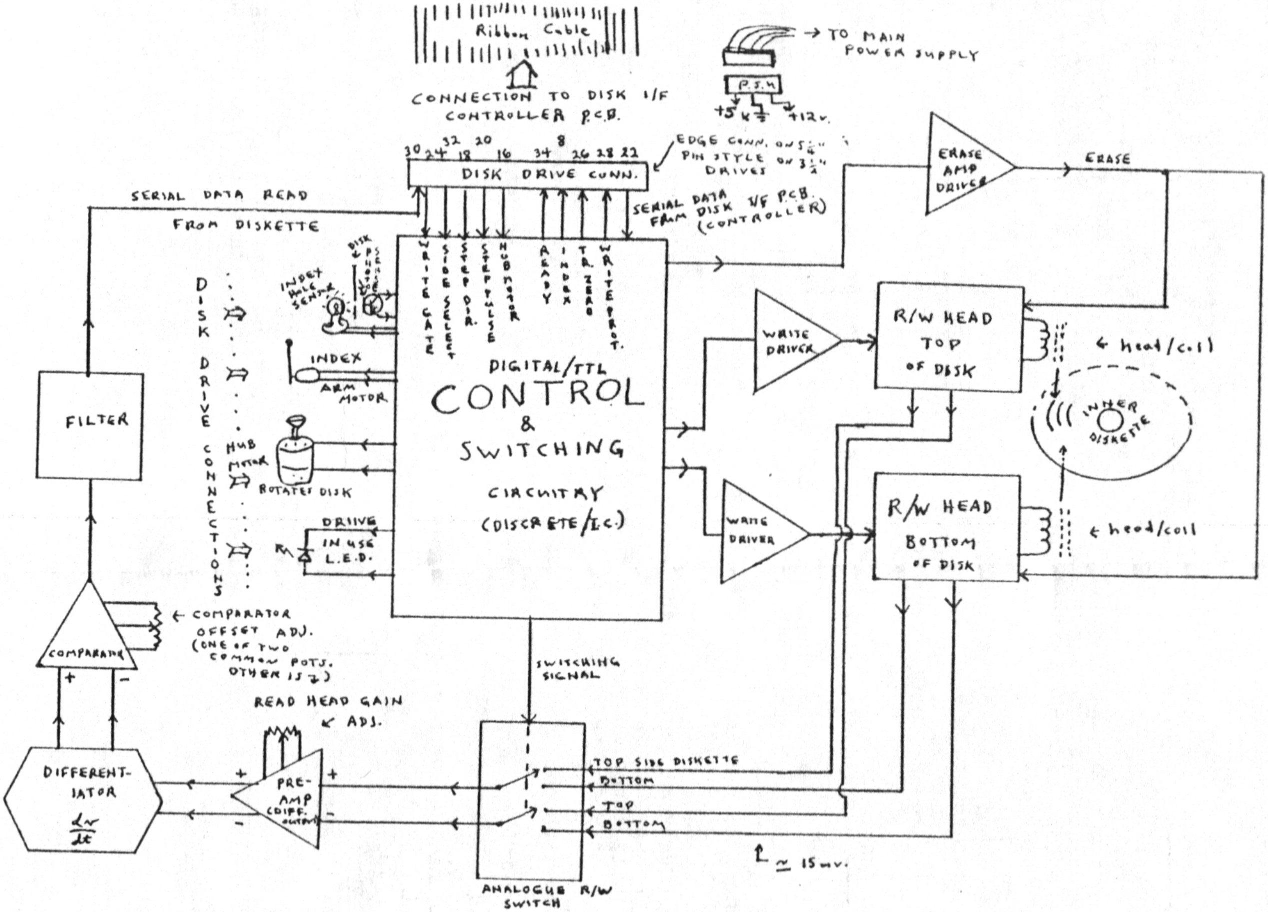


Fig. World's Cheapest Drawing Board - But A Good One. Cost $15-30 compared to $75-500+

Experimenting with CAD, computer drafting is another topic to deal with, some day, although it has not made hand drafting obsolete for small jobs.



NOTE: → ← ARROWS SHOW WHETHER INPUT OR OUTPUT, NOT CURRENT DIR'N.

With a disk drive mechanism comes an electronic circuit board to do Analogue/Digital conversion and switching. It takes digital signals from the computer's disk drive interface board and converts them to the necessary analogue, measured voltages and currents to run the mechanism and the read/write heads for the magnetic medium. This diagram shows its works.

TYPICAL DISK DRIVE
P.C.B. ELECTRONICS
BLOCK FUNCTION DIA.

1989, TJ BULLETIN A.P.
REV. 1991

FROM "ZX-81 UNOFFICIAL LDOS MANUAL"